DEFINE WORK FILE

DEFINE WORK FILE *n operand1* **[TYPE** *operand2* **]**

Operand	Possible Structure					Possible Formats										Referencing Permitted	Dynamic Definition
Operand1	C	S				A										yes	no
Operand2	C	S				A										yes	no

Related Statements: WRITE WORK FILE | READ WORK FILE | CLOSE WORK FILE

Function

The DEFINE WORK FILE statement is used to assign a file name to a Natural work file number within a Natural application.

This allows you to make or change work file assignments dynamically within a Natural session or overwrite work file assignments made at another level.

When this statement is executed and the specified work file is already open, the statement will implicitly cause that work file to be closed.

Work File Number - n

n is the work file number (1 to 32). This is the number to be used in a WRITE WORK FILE, READ WORK FILE or CLOSE WORK FILE statement.

Work File Name - operand1

operand1 is the name of the work file.

- Work File Name on UNIX and Windows
- Work File Name on Mainframe Computers

Work File Name on UNIX and Windows

The file name (*operand1*) may contain environment variables.

If a file with the specified name does not exist, it will be created.

Under UNIX and Windows, it is possible to use physical work file names.

Work File Name on Mainframe Computers

As *operand1* you specify the name of the dataset to be assigned to the work file number.

Operand1 can be 1 to 253 characters long. You can specify either a logical or a physical dataset name.

Note for CMS:
For work files assigned to the CMS work file access method (see NTWORK profile parameter), operand1 must be omitted.

Work File Type - *operand2*

Operand2 specifies the type of work file.

The following is a list of work file types:

- DEFAULT
- TRANSFER
- SAG
- ASCII
- ASCII-COMPRESSED
- ENTIRECONNECTION
- UNFORMATTED
- PORTABLE
- FORMATTED

DEFAULT

Determines the file type from the extension for upward compatibility.

TRANSFER

Is used to transfer data to and from a PC with Entire Connection.

This work file type represents a data connection between a Natural session on UNIX and a Entire Connection terminal on a PC. The work file data is written in Entire Connection format on the PC.

Note:
This format cannot be used under Windows.

SAG

This is the binary format.

ASCII

Files in ASCII are "text" files with records terminated by [a carriage return] linefeed.

ASCII-COMPRESSED

Is a file in ASCII format, with the exception that all trailing blanks are removed.

ENTIRECONNECTION

With this work file type, you can read and write (using the statements READ and WRITE, for example) directly to a work file in Entire Connection format on the local disc.

Note:
This work file type is available on PCs and on UNIX. No transfer to PC is possible. The Entire Connection terminal is not used in this process.

UNFORMATTED

Is a completely unformatted file. No formatting information is written (neither for fields nor for records).

PORTABLE

files which can handle dynamic variables exactly and can also be transported: e.g., from a little endian machine to a big endian machine, and vice versa

FORMATTED

(Mainframe only) regular record-oriented work files

The value of *operand2* is handled in a case insensitive way and must be enclosed in quotes or provided in an alphanumeric variable.
Examples:

```
DEFINE WORK FILE 17 #FILE TYPE 'unformatted'
  #TYPE := 'SAG'
  DEFINE WORK FILE 18 #FILE TYPE #TYPE
```

Note for mainframes:
The only TYPEs accepted by Natural for mainframes are FORMATTED and UNFORMATTED: FORMATTED defines a regular record-oriented work file, which is subject to the same handling as in previous Natural versions. UNFORMATTED treats a work file as a byte-stream with no record boundaries. Note that type UNFORMATTED will be rejected by Entire Connection.

For more information on work files, see the section Work File Formats.

Work File Name under OS/390

Under OS/390, for a work-file number that is defined with the access method AM=STD (whether automatically in the JCL, in the NTWORK macro of the Natural parameter module or dynamically using the profile parameter WORK), *operand1* can be:

- a logical dataset name (DD name, 1 to 8 characters);
- a physical dataset name of a cataloged dataset (1 to 44 characters) or a physical dataset member name;
- a path name and member name of an HFS file (1 to 253 characters) in an MVS UNIX Services environment;
- a JES spool file class;
- "NULLFILE" (to indicate a dummy dataset).

- Logical Dataset Names
- Physical Dataset Names
- HFS Files
- PFS Files
- JES Spool File Class
- NULLFILE
- Allocation and De-Allocation of Datasets
- Work Files in Server Environments
- Further Information

Logical Dataset Names

For example:

```
DEFINE WORK FILE 21 'SYSOUT'
```

The specified dataset must have been allocated before the DEFINE WORK FILE statement is executed.

The allocation can be done via JCL, CLIST or dynamic allocation (SVC 99). For dynamic allocation you can use the user exit USR2021 in library SYSEXT.

The dataset name specified in the DEFINE WORK FILE statement overrides the name specified with the subparameter DEST of the NTWORK macro or WORK profile parameter.

Optionally, the dataset name may be prefixed by "DDN=" to indicate that it is a DD name. For example:

```
DEFINE WORK FILE 22 'DDN=XYZ'
```

Physical Dataset Names

For example:

```
DEFINE WORK FILE 23 'TEST.WORK.FILE'
```

The specified dataset must exist in cataloged form. When the DEFINE WORK FILE statement is executed, the dataset is allocated dynamically by SVC 99 with the current DD name and option DISP=SHR.

If the dataset name is 8 characters or shorter and does not contain a period ".", it might be misinterpreted as a DD name. To avoid this, prefix the name with "DSN=". For example:

```
DEFINE WORK FILE 22 'DSN=WORKXYZ'
```

If the dataset is a PDS member, you specify the PDS member name (1 to 8 characters) in parentheses after the dataset name (1 to 44 characters). For example:

```
DEFINE WORK FILE 4 'TEST.WORK.PDS(TEST1)'
```

If the specified member does not exist, a new member of that name will be created.

HFS Files

For example:

```
DEFINE WORK FILE 14 '/u/nat/rec/test.txt'
```

The specified path name must exist. When the DEFINE WORK FILE statement is executed, the HFS file is allocated dynamically. If the specified member does not exist, a new member of that name will be created.

For the dynamic allocation of the dataset, the following OS/390 path options are used:

```
PATHOPTS=(OCREAT,OTRUNC,ORDWR)
  PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
  FILEDATA=TEXT
```

When an HFS file is closed, it is automatically de-allocated by OS/390 (regardless of the setting of the subparameter FREE in the NTWORK macro or WORK profile parameter).

To read an HFS file, you have to use the user exit USR2021 instead of the DEFINE WORK FILE statement, because of the OTRUNC option. This option will reset the HFS file at the first read access and result in an empty file.

PFS Files

PFS files are available under Com-plete (SMARTS AM=SMARTS). Any printer name can be assigned, even if it has not been defined to Natural.

For example:

```
DEFINE PRINTER (14) OUTPUT '/nat/path/workfile'
DEFINE PRINTER (14) OUTPUT 'workfile'
```

It depends on the MOUNT_FS parameter of SMARTS whether the file is located on a SMARTS portable file system or on the native file system. The first element of the path (/nat/) determines the target file system.

If the string does not start with a slash "/", the path of the file is taken from the environment variable $NAT_WORK_ROOT.

The specified path name must exist. When the DEFINE PRINTER statement is executed, the file is allocated dynamically. If the specified member does not exist, a new member of that name will be created.

JES Spool File Class

To create a JES spool dataset, you specify SYSOUT=x (where x is the desired spool file class). For the default spool file class, you specify SYSOUT=*.

Examples:

```
DEFINE WORK FILE 10 'SYSOUT=A'
  DEFINE WORK FILE 12 'SYSOUT=*'
```

To specify additional parameters for the dynamic allocation, use the user exit USR2021 in the library SYSEXT instead of the DEFINE WORK FILE statement.

NULLFILE

To allocate a dummy dataset, you specify NULLFILE as *operand1*:

```
DEFINE WORK FILE n 'NULLFILE'
```

This corresponds to the JCL definition:

```
// DD-name DD DUMMY
```

Allocation and De-Allocation of Datasets

When the DEFINE WORK FILE statement is executed and a physical dataset name, HFS file, spool file class or dummy dataset has been specified, the corresponding dataset is allocated automatically. If the logical file is already open, it will be closed automatically, except when the profile parameter CLOSE=FIN has been specified, in which case an error will be issued. Moreover, an existing dataset allocated with the same current DD name is automatically de-allocated before the new dataset is allocated. Work files that are to be allocated dynamically have to be predefined in the Natural parameter module with AM=STD.

To avoid unneccessary overhead by unsuccessful premature opening of work files not yet allocated at the start of the program, work files should be defined with the subparameter OPEN=ACC (open at first access) in the NTWORK macro or WORK profile parameter.

In the case of an HFS file, or a work file defined with the subparameter FREE=ON in the NTWORK macro or WORK profile parameter, the work file is automatically de-allocated as soon as it has been closed.

As an alternative for the dynamic allocation and de-allocation of datasets, the user exit USR2021 in the library SYSEXT is provided. This user exit also allows you to specify additional parameters for dynamic allocation.

Work Files in Server Environments

In server environments, errors may occur if multiple Natural sessions attempt to allocate or open a dataset with the same DD name. To avoid this, you either specify the work file with the subparameter DEST=* in the NTWORK macro or WORK profile parameter, or you specify DEFINE WORK FILE '*' in your program before the actual DEFINE WORK FILE statement; Natural then generates a unique DD name at the physical dataset allocation when the first DEFINE WORK FILE statement for that work file is executed.

All work files whose DD names begin with "CM" are shared by all sessions in a server environment. A shared work file opened for output by the first session is physically closed when the server is terminated. A shared work file opened for input is physically closed when the last session closes it, that is, when it receives an end-of-file condition. When a work file is read concurrently, one file record is supplied to one READ WORK FILE statement only.

Further Information

For information on work files, see also Operations for Mainframes.

Work File Name under VM/CMS

Under VM/CMS (for work files defined in the Natural parameter module with AM=STD), the same applies to *operand1* as under OS/390 (see above) but with the following differences:

- Instead of dynamic allocation via MVS SVC 99, the CMS command FILEDEF is used to define a file.
- HFS files are not supported.
- JES spool classes are not supported.
- In addition, the following syntax is used:

  ```
  DEFINE WORK FILE n 'fname ftype fmode (options)'
  ```

  This generates the CMS command:

  ```
  FILEDEF ddname-n DISK fname ftype fmode (options)
  ```


- Moreover, the following syntax is allowed:

  ```
  DEFINE WORK FILE n 'FILEDEF=filedef-parameters'
  ```

  This generates the CMS command:

  ```
  FILEDEF ddname-n =filedef-parameters
  ```

  For example:

  ```
  DEFINE WORK FILE 5 'FILEDEF=TAP1 SL 2 VOLID BKUP08 (BLKSIZE 20000)'
  ```

  This generates the CMS command:

  ```
  FILEDEF CMWKF05 TAP1 SL 2 VOLID BKUP08
  ```


Work File Name under BS2000/OSD

Under BS2000/OSD, for a work-file number that is defined with the access method AM=STD (whether automatically in the JCL, in the NTWORK macro of the Natural parameter module or dynamically using the profile parameter WORK), you can use *operand1* to specify a file name or a link name that is allocated to this work file.

In this case, *operand1* can have a length of 1 to 253 characters and one of the following meanings:

- a BS2000/OSD link name (1 to 8 characters)
- a BS2000/OSD file name (9 to 54 characters)
- a generic BS2000/OSD file name (wildcard)
- a BS2000/OSD file name and link name
- a generic BS2000/OSD file name and link name (wildcard)
- *DUMMY

The following rules apply.

1. File name and link name can be specified as positional parameters or keyword parameters. The corresponding keywords are **FILE=** and **LINK=**. Mixing positional and keyword parameters is allowed but not recommended.
2. A string with a length of 1 to 8 characters without commas is interpreted as a link name. This notation is compatible with earlier versions of Natural.

   Example:

   ```
   DEFINE WORK FILE 1 'W01'
   ```

   The corresponding definition with a keyword parameter is:

   ```
   DEFINE WORK FILE 1 'LINK=W01'
   ```

3. A string of 9 to 54 characters without commas is interpreted as a file name.

   Example:

   ```
   DEFINE WORK FILE 2 'NATURAL31.TEST.WORKFILE02'
   ```

   The corresponding definition with a keyword parameter is:

   ```
   DEFINE WORK FILE 2 'FILE=NATURAL31.TEST.WORKFILE02'
   ```

4. The following input is interpreted without considering the length and therefore forms exceptions to Rules 2 and 3:
   - keyword input: LINK=, FILE=
   - *DUMMY
   - NULLFILE (equivalent to *DUMMY)
   - *
   - *,*

   Example: DEFINE WORK FILE 7 'FILE=Y' is a valid file allocation and not a link name, although the string of characters contains fewer than 9 characters.

5. Generic file names are formed as follows:

   Wnn.userid.tsn.date.time.number

   where
   nn is a work-file number
   userid is a Natural user-ID, 8 characters

tsn is the BS2000/OSD TSN of the current task, 4 digits
date is DDMMYYYY
time is HHIISS
number is a number, 5 digits

6. Generic link names are formed as follows:

   NWFnnnnn

   nnnnn is a 5-digit number that is increased by one after every generation of a dynamic link name.

7.

   Changing the file allocation for a work-file number causes an implicit CLOSE of the work file allocated so far.

You are strongly recommended, in all cases except when you only specify a link name (for example: W01), to work with keyword parameters. This avoids conflicts of interpretation with additional reports and is essential for file names with fewer than 9 characters.

Example:

```
DEFINE WORK FILE 3 'LINK=#W03'
  DEFINE WORK FILE 3 'FILE=#W03'
```

Link Name

Example:

```
DEFINE WORK FILE 1 'LINKW01'
```

means the same as

```
DEFINE WORK FILE 1 'LINK=LINKW01'
```

A file with the LINK 'LINKW01' must exist at runtime. This can be created either using JCL before starting Natural or by dynamic allocation from the current application. For dynamic allocation, the user exit USR2029 in the library SYSEXT can be used. If, before execution, the link was active on another file, for example: 'W01', this will be released or retained depending on the value of the profile parameter FREE (possible values are ON and OFF). Release is done via an explicit RELEASE call to the BS2000/OSD command processor.

File Name

Example:

```
DEFINE WORK FILE 2 'NATURAL31.TEST.WORK02'
```

means the same as

```
DEFINE WORK FILE 2 'FILE=NATURAL31.TEST.WORK02'
```

The file specified in *operand1* is set up using a FILE macro call and inherits the link name that was valid for the corresponding work file before execution of the DEFINE WORK FILE statement.

Generic File Name

Example:


```
DEFINE WORK FILE 21 '*'
```

means the same as

```
DEFINE WORK FILE 21 'FILE=*'
```

A file with a name created according to Rule 4 is set up using a FILE macro call and inherits the link name that was valid for the corresponding work file before execution of the DEFINE WORK FILE statement.

```
DEFINE WORK FILE 22 'FILE=*,LINK=WFLK22'
```

A file with a name created according to Rule 4 is set up with the specified link name, using a FILE macro call.

File Name and Link Name

Example:

```
DEFINE WORK FILE 11 'NATURAL31.TEST.WORKF11,LNKW11'
```

means the same as

```
DEFINE WORK FILE 11 'FILE=NATURAL31.TEST.WORKF11,LINK=LNKW11'
```

which means the same as

```
DEFINE WORK FILE 11 'FILE=NATURAL31.TEST.WORKF11,LNKW11'
```

The file given in *operand1* is set up with the specified link name, using a FILE macro call and allocated to the corresponding work-file number.

Generic File Name and Link Name

Example:

```
DEFINE WORK FILE 27 '*,*'
```

means the same as

```
DEFINE WORK FILE 27 'FILE=*,LINK=*'
```

A file with a file name and link name created according to Rule 4 and Rule 5 is set up using a FILE macro call and allocated to the specified work file 27.

Note:
When file name and link name are specified, the previous link name is not released, regardless of the value of the profile parameter FREE.

